(51) International Patent Classification 6 : G06F 12/08	A1	(11) International Publication Number: **WO 97/04392** (43) International Publication Date: 6 February 1997 (06.02.97)

(71) Applicant (for all designated States except US): INTER-NATIONAL BUSINESS MACHINES CORPORATION [US/US]; Old Orchard Road, Armonk, NY 10504 (US).

(72) Inventors; and
(75) Inventors/Applicants (for US only): GETZLAFF, Klaus, Jörg [DE/DE]; Friesenweg 26, D-71101 Schönaich (DE). WILLE, Udo [DE/DE]; Wengertsteige 47, D-71088 Holzgerlingen (DE). DOETTLING, Gerhard [DE/DE]; Karlstrasse 22, D-72135 Dettenhausen (DE). LEPPLA, Bernd [DE/DE]; Koenigstrasse 93, D-71139 Ehningen (DE). TAST, Hans-Werner [DE/DE]; Hartmannstrasse 66, D-71093 Weil im Schönbuch (DE). MAK, Pak-kin [US/US]; 13 Ridgenwoodterrace, Poughkeepsie, NY 12603 (US). JACKSON, Kathy, M. [US/US]; 21 Greenbush Drive, Poughkeepsie, NY 12601 (US). SHEN, William, W. [US/US]; 18 Kelterhouse Drive, Poughkeepsie, NY 12603 (US). LANGSTON, Keith [US/US]; 114 Hardenburgh Road, Ulster Park, NY 12487 (US).

(74) Agent: SCHÄFER, Wolfgang; IBM Deutschland Information-ssysteme GmbH, Patentwesen und Urheberrecht, D-70548 Stuttgart (DE).

(54) Title: SHARED CACHE MEMORY DEVICE
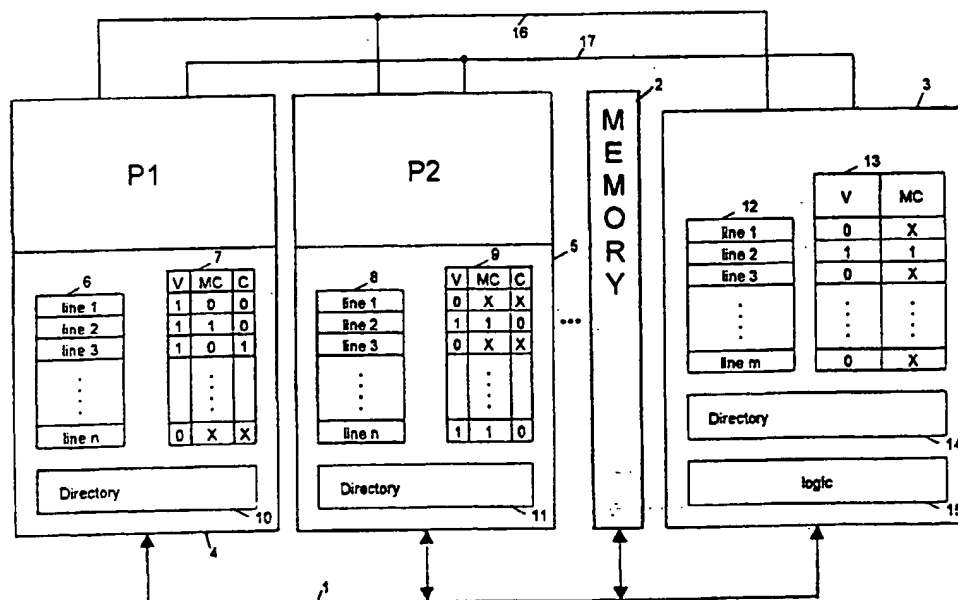
(57) Abstract

The invention relates to a cache memory device for usage as a shared cache (3) in a multiprocessor system. The cache memory device (3) is organized in a plurality of storage blocks and has storage means (13) for storage of status information being indicative of valid or shared data of the storage blocks. The cache memory (3) further comprises logic means (15) to selectively store data in one of the storage blocks and to set the status information correspondingly.

DESCRIPTION

SHARED CACHE MEMORY DEVICE

The invention relates to a cache memory device for usage
as a shared cache in a multiprocessor system, wherein the
memory space is organized in a plurality of storage
blocks. Furthermore, the invention relates to a computer
system incorporating such a multiprocessor system and to
a method to operate a shared cache memory device in such
a multiprocessor system.

From US-A-5 025 365 a snooping coherency protocol for a
multiprocesor network is known. Every processor has its
own private cache and bus interface means and the network
is connected via a common system bus. Each processor has
its own cache directory and image directory that
duplicate each other non-atomically. The snooping
protocol utilizes the duality of directories coupled with
the non-atomicity of directory updates to maximize
processor-cache availability and minimize processor-cache
access times thus supporting high performance processors.

In EP-A-0 349 123 a multi-processor computer system
having shared memory and private cache memories is
disclosed. As it is generally the case this multi-
processor system is implemented using a common system bus
as the communication mechanism between CPU, memory, and
I/O adapters. It is also common to include features on
each CPU module, such as a cache memory, that enhance the
performance of the execution of instructions in the CPU.
Many architectures require that the hardware employs a
mechanism by which the data in the individual CPU cache
memories is kept consistent with data in main memory and
with data in other cache memories. One such method
involves each CPU monitoring transactions on the system

bus, and taking appropriate action when a transaction
appears on the bus which would render data in the CPU's
cache incoherent. If the CPU uses queues to hold records
of incoming transaction information until it can service
them, the bus interface must guarantee that the queued
items are processed by the cache in the correct order. If
this is not done, certain types of shared data protocols
fail to operate correctly.

The EP-A-0 349 123 describes a method by which hardware
can guarantee the serialization of transactions requiring
service by the CPU cache. The serialization method
described guarantees that shared memory protocols operate
correctly.

In known signal processor systems the processor transfers
instructions and data from system memory to the cache
memory in order to have quick access to the variables of
the currently executing program. As additional data, not
in the cache, is required, such data is transferred from
the main memory by replacing selected data in the cache.
Various techniques or algorithms are utilized to
determine which data is replaced. Since the data in the
cache is duplicative of data in the main memory as seen
from other processors, changes to data in one memory must
similarly be changed or noted in the other memory. The
problem of maintaining consistency between the cache and
main memory data is referred to as coherency. For
example, if the data in the cache is modified, the
corresponding data in the main memory must similarly be
modified. A scheme for modifying data in the main memory
at the same time as data in the cache is known as a
write-through cache. Alternatively, if only the cache
memory data is modified, then such modified cache data
must overwrite the main memory data when it is replaced.

In many computer systems, typically large systems, the main memory may be accessed by devices other than the system processor. For example, data my be written to the main memory by another device while the processor is operating on data in its cache. If the data written to main memory overwrite data presently in the cache, a coherency problem is presented. It is known in the art to use a method such as bus snooping to address this problem. This involves monitoring a bus for writes to main memory and then checking a tag directory associated with the cache to see if the data is in the cache. If present, a flag is set in the cache tag directory to invalidate the entry so that the old data is not used by the processor.

Typically, in multiprocessor systems, the processing units, main memory peripheral devices and other devices are all tied together by a multidrop system bus which carries messages, instructions, data, addresses, protocol signals, etc. to and from all devices. In multiprocessor systems having large memory capacity, the main memory is constructed of inexpensive, slow memory components. Each access to main memory to obtain data requires use of the data and address portions of the system bus and a substantial amount of memory access time thus seriously inhibiting the performance of the computer system. It is found that, in computer systems having large main memory capacity, a substantial portion of the memory is used very infrequently. On the other hand, certain data blocks, i.e. memory locations are used very frequently by the processing units (or processors).

Because of this characteristic, many large memory capacity computer systems employ small volume, high-speed memory buffers or caches for storing copies of the frequently used data locally to the processing units. By

placing the frequently used data in a faster local memory
cache, processing time and use of the system bus can be
significantly reduced. When a processing unit needs to
read data from an address, it accesses the high speed
data cache. If the requested data address is not in the
data cache, only then is an access to slower main memory
over the system bus executed. To make the addition of a
cache memory efficient, the access time of the cache
memory should be about five to fifteen times faster than
the access time of the main memory.

It is further found that the most recently accessed data
is more likely to be accessed again by the processing
units than that data which has not been accessed
recently.

The data cache is inserted between the processing unit
and the system bus which couples between processing
units, the main memory and all other devices on the
system. Typically, some form of virtual addressing is
used to address the space in the data cache.

The operation of the data cache during a read instruction
from the processing unit is straightforward. If the
requested data is in the cache, it is forwarded directly
to the processing unit. If, however, the data is not in
the cache, the cache obtains the data from main memory,
stores it in the cache and also forwards it to the
processing unit. In the case of a write to memory, there
are various possible implementations. In one possible
method, data can be written to the cache and the main
memory simultaneously. However, a substantial amount of
processor time is saved by only writing to the cache
memory during the write operation and providing a flag
for the location in the cache which indicates that the
data in that location has been updated and is no longer

consistent with the data in main memory. The main memory, then, can be updated only when this block of memory is to be removed from the cache to make way for a new block. This method saves unnecessary write operations to the main memory when a given cache word is updated a number of times before it is replaced in the cache.

Today there are numerous multiprocessor systems, i.e. computer systems having multiple processing units, available on the market. In multiprocessor systems using memory caches, each processor may have its own dedicated memory cache or they may all share a single memory cache. The use of data caches in these systems is more complex than that previously described due to the need to service multiple processors, and, if each processor has its own data cache, maintain consistency of data between multiple caches. Even further, some manufacturers now produce computer systems having two levels of cache memory: a small, very fast primary cache and a larger and frequently slightly slower secondary cache.

Various prior art approaches to the design of system buses and system bus protocols in multicomputer systems are discussed in J. Archibald & J. Baer, Cache Coherence Protocols: Evaluation Using A Multiprocessor Simulation Model, ACM Transactions on Computer Systems, Vol. 4, No. 4 (November 1986).

A further concept to maintain data consisentcy is known from an article entitled "Data Consistency In A Multiprocessor System With Store-In Cache Concept", microprocessing and microprogramming 32 (1991), 215 -220, North-Holland by G. Doettling and US-A-5 113 514. The approach described in the article by G. Doettling, which is also known as the MESI-protocol is also employed in the multiprocessor system known from EP-A-0 575 651 and

WO-A 95/03568.

The MESI-protocol has four states: Modified, Exclusive,
Shared and Invalid. A modified cache line is a cache line
which has been changed by one of the processors which
holds the actual copy in its cache. An exclusive cache
line is a cache line which is owned by only one processor
which may write into this line. A shared cache line is a
cache line which is copied in the caches of more than one
processor. An invalid cache line is an invalid copy of a
cache line.

These cache line states may be recorded in each private
cache directory using three bits: A multiple copy bit
(NC-bit), a change bit (C-bit) and valid bit (V-bit). A
bus snooping action is initiated in each processor than a
command and its address are placed on the bus. The
command and its address are examined by the snooping
processors and the required actions are taken. Each of
the processors searches its cache directory for the
address. A cache line hit, i.e. match of the address on
the bus and an address stored in the directory, is
signaled by the processor which detected the cache line
hit in its cache directory to the other processors. If
this cache line where the hit occurred is valid and
changed this cache line represents the actual copy in the
system. In this case the actual cache line is
retransmitted to the main memory. This is also called
"Cast-Out". The cache line status in the cache
directories of the caches which also have a cache line
hit is updated correspondingly.

This snooping action requires a couple of cycles to
interrogate the bus address, the directory and signaling
the result to the other processors. It is normally a
synchronous operation.

It is an object of the present invention to provide an
improved cache memory device for usage as a shared cache
in a multiprocessor system, an improved multiprocessor
system and an improved computer system incorporating such
a cache memory device and an improved method to operate a
shared cache memory device in a multiprocessor system.
The object of the invention is solved by the features
laid down in the independent claims.

The cache memory device of the invention is organized in
a plurality of storage blocks, such as storage lines. The
storage blocks can be used for the virtual addressing of,
the space in the cache memory device. The cache memory
device has storage means for storage of status
information for each of the storage blocks. Such a
storage means can be realized in the cache memory device
by a dedicated storage space wherein such status
information can be stored for each of the storage blocks.
The status information is indicative of valid or shared
data which is stored or which is to be stored in the
storage block to which the respective status information
belongs. The cache memory device further comprises logic
means to selectively store data in the storage blocks and
to set the status information correspondingly. The
provision of status information being indicative of
shared data is beneficial because this allows to take
full advantageous of the shorter access time of the
shared cache memory device as compared to the private
caches of the processors of the multiprocessor system.
If one of the processors requests data which are present
in a storage block of the shared cache memory device and
if the status of these data is "shared", the share memory
device can output the data stored in that block on the
bus without further access time penalty, i.e. without
having to wait for the other processors to signal that
these data required by the requesting processor is also

present in one or more of the private cache memories of
the snooping processors or not. This is because no cast
out of the requested data from the private caches of the
snooping processors can occur by definition of the
protocol: if the status of the requested data is valid
and shared in the cache memory device this must also be
the case for any other additional copy of these data in
the private caches of the snooping processors since
otherwise the consistency of the data stored in the
private caches and the shared cache memory device could
not be maintained.

If during operation of the multiprocessor system one of
the processors fetches valid data, these data are also
stored in the shared cache memory device. This is also
done if changed data is cast out by one of the snooping
processors to update the main memory. In this case the
shared memory device stores the updated copy of the
changed data in the corresponding storage block. This is
controlled by logic means, i.e. a number of logic gates,
which are incorporated in the cache memory device.

According to a preferred embodiment of the invention the
MESI-protocol is used to maintain the consistency of the
data being stored in the private caches and the shared
cache memory device. In this case the status information
for each of the storage blocks of the shared cache memory
device comprises a valid bit (V-bit) and a multiple copy
bit (MC-bit). If the multiple copy bit of one of the
storage blocks is logically one this indicates that the
data of that storage block is shared, i.e. at least two
private caches of different processors have a copy of
these data stored therein.

According to a preferred embodiment of the invention the
status information being indicative of shared data in the

shared cache memory device is already set to logically
one if data which generally remains unchanged such as
instructions, is requested by one of the processors. This
is irrespective of the requested data already being
shared by one of the private caches of the other
processors or not. It is assumed that data, such as
instructions, generally remains unchanged. Generally it
is implied that a situation in which one of the private
caches holds valid data which is changed - causing a
cast-out situation - will occur only rarely. Hence it is
possible in most cases to take full advantage of the
shorter access time of the shared cache memory device to
respond to a second request by another processor for the
same data.

In the following a preferred embodiment of the invention
is explained in more detail with reference to the drawing
in which:

Fig. 1   is a schematic block diagram of a
           multiprocessor system and a shared cache memory
           device according to the invention;

Fig. 2   is a signal diagram of the multiprocessor
           system when no usage is made of the shared
           cache memory device;

Fig. 3   is a signal diagram of the multiprocessor
           system when usage is made of the shared memory
           device.

The multiprocessor system of Fig.1 comprises a plurality
of processors P1, P2, ..., P_n. Only two of the processors
- P1 and P2 - are shown in Fig.1 by way of example. The
processors of the microprocessor system communicate with
each other and with the main memory 2 of the system and

the shared cache memory 3 via the system bus 1. Each of
the processors has a private cache memory. The processors
P1 and P2 have the private cache memories 4 and 5,
respectively. Each of the private cache memories has a
storage array for the storage of a plurality of storage
blocks. In this case the storage blocks are realized as
storage lines i.e. lines 1 to n. Furthermore, each of the
private caches has a storage space for storage of a valid
bit (V-bit), a multiple copy bit (MC-bit) and a change
bit (C-bit) for each of the lines. In the example shown
in Fig.1 the first line of the storage space 7 holds V-,
MC- and C-bit for the storage line 1 of the storage array
6. The same applies analogously for the further lines 2
to n. In this example the V-, MC- and C-bit of line 1
equals logically 1, 0, 0, respectively. The corresponding
bits for line 2 are 1, 1, 0; for line 3 1, 0, 1 and for
line n 0, X, X. "X" indicates that this bits position is
a "don't care" bit. This is the case if the valid bit is
logically 0.

The corresponding bits of the storage line 1 of the
storage array 8 of the private cache 5 are 0, X, X, and
for the further lines 2 to n: 1, 1, 0; 0, X, X; ...; 1,
1, 0. These bits are stored in the storage space 9 of the
private cache 5. Each of the private caches has a
directory which holds the logical addresses of the data
which are stored in the storage lines. For example, the
directory 10 of the private cache 4 holds the logical
address of the data stored in the storage line 1 as well
as the further addresses of the data stored in the
further lines 2 to n. The same applies analogously to the
directory 11 of the private cache 5 and to the
directories of the other private caches of the other
processors of the multiprocessing system which are not
shown in the drawing.

The shared cache memory device 3 has a storage array 12
for the storage of the storage lines 1 to m. The number m
of storage lines which can be stored in the shared cache
memory 3 generally is much greater than the number n of
storage lines of the private cache memories. However the
number of bit positions in a storage line of the storage
array 12 of the shared cache memory 3 equals the number
of bit positions in a storage line of one of the private
caches. This is essential for the method of addressing
which is employed in this example. The shared cache
memory device 3 further comprises a storage space 13
which serves to store the status information for each of
the storage lines of the storage array 12. The storage
space 13 has two bits of status information for each of
the storage lines of the storage array 12, i.e. a V- and
a MC-bit. In the example shown in Fig.1 the storage line
1 of the storage array 12 has a valid bit V which equals
logically 0; as a consequence the multiple copy bit MC is
"don't care" (X). The corresponding status information of
the lines 2 to m is 1,1; 0,X; ...; 0,X. Furthermore the
shared cache memory device 3 comprises a directory 14
which holds the logical addresses of the data stored in
the storage lines 1 to m in the storage array 12, i.e.
the logical address of the data stored in the storage
line 1 and for the further storage lines 2 to m,
respectively. The operation of the shared cache memory
device 3 is controlled by the logic 15. The logic 15
enables the storage of data in a selected one of the
storage lines of the storage array 12 and sets the status
information in the storage space 13 correspondingly.

The processors P1, P2, ..., P_n as well as the shared
cache memory 3 are interconnected by the signal lines 16
and 17. If the signal line 16 is raised by one of the
processors this causes a bus snooping operation of the
other processors and of the shared cache memory 3. The

bus snooping of the shared cache memory 3 is also
accomplished by the logic 15.

The signal line 17 is raised by a snooping processor
which has detected a hit, or with other words an address
match in its directory. If the signal line 17 is raised
and a cast-out is required then this signal remains
active until the cast-out operation is completed.

In table 1 eight examples are shown for possible states
of the status information in a requesting processor and
in a snooping processor according to the MESI-protocol
which is employed here in order to maintain consistency
of the data stored in the caches. In the first group of
examples (examples 1. to 4.) the requesting processor
requires a sequence of data having the logical address i
for storage in one of the lines, for example line j of
its storage array 6 for a read reference. This is called
line fetch due to fetch because the requesting processor
P1 does only fetch the required sequence of data. As
opposed to this in the examples 5 to 8. "line fetch due
to store" the requesting processor requires the sequence
of data for write purposes.

TABLE 1

	Requesting processor	---- Snooping processor ----		
Bus operation	Cache line New state	Cache line Initial state	Cache line New state	Action
	V MC C	V MC C	V MC C	
Line Fetch due to fetch	1. 1 0 0 2. 1 1 0 3. 1 1 0 4. 1 1 0	0 0 0 1 0 0 1 1 0 1 0 1	0 0 0 1 1 0 1 1 0 1 1 0	Cast-Out
Line Fetch due to store	5. 1 0 1 6. 1 0 1 7. 1 0 1 8. 1 0 1	0 0 0 1 0 0 1 1 0 1 0 1	0 0 0 0 0 0 0 0 0 0 0 0	Cast-Out

(V = Valid / MC = Multiple Copy / C = Changed)

The initial situation is that the data having the logical address i are not present in the private cache of the requesting processor or if the data having the logical address i is present in the private cache the V-bit is logically 0.

As a consequence the requesting processor puts a line fetch command together with the logical address i onto the bus and initiates a bus snooping operation of the other processors P2 to PN and the shared cache memory 3. The bus snooping is initiated by the processor P1 over the signal line 16.

In the first example the snooping processor considered
here has no logical address in its directory which
matches the logical address i of the requested line of
data or - if such a match should occur - the
corresponding data is not valid which is indicated by the
V-bit which equals logically 0. In this case no action or
change of the status information is required by this
snooping processor.

For the following examples 2., 3. and 4. it is assumed
that the directory of the snooping processor considered
here holds a logical address which matches the logical
address i of the requested data. In the second example
the corresponding line of data has the following status
information:

The V-bit is 1 the MC-bit is 0 and the C-bit is also 0.
This means that the snooping processor has a valid copy
of the requested line of data, that this line of data is
unchanged with respect to the corresponding data of the
same logical address i being stored in the main memory 2
and that there are no further copies in other snooping
processors which is indicated by the MC-bit. In this case
the requested line of data is outputted by the main
memory 2 or by the shared cache memory 3 onto the system
bus 1 to be inputted to the private cache of the
requesting processor P1. As a result, the requesting
processor and the snooping processor hold a valid copy of
the line of data having the logical address i. This is
reflected by the multiple copy bit (MC-bit). In addition
the shared memory cache 3 will set its MC-Bit to 1 if not
already done by a previous bus operation.

The multiple copy bit is set to be logically 1 in the
line of the storage space 7 of the processor P1 which
belongs to the storage line in the storage array 6 into

which the requested data is stored. The same applies
analogously for the multiple copy bit which belongs to
the storage line in the storage array of the snooping
processor which holds the requested data having the
address i. Hence the status information as indicated by
the V-, MC- and C-bits is 1,1,0 in the requesting
processor and the snooping processor in which the hit
situation occurred.

Example 3. differs from example 2. in that the multiple
copy bit initially equals logically 1 in the snooping
processor. This means that there is more than one copy in
the private cache memories of the processors. In this
case no action has to be performed by the snooping
processor considered here.

In the example 4. the initial state of the status
information in the snooping processor is 1,0,1, i.e. the
snooping processor holes a valid copy of the requested
data but this copy has been changed previously so that
the C-bit is logically 1. This means that this copy which
is stored in the private cache of this snooping processor
does no longer equal the corresponding line of data
having the logical address i which is stored in the main
memory 2. In order to prevent the requesting processor P1
to get an invalid copy of the data from the main memory 2
the snooping processor has to cast-out the changed line
of data. Therefore the snooping processor has to perform
the following actions:

The changed line of data having the address i is
outputted by the snooping processor from its private
cache to the memory 2 via the system bus 1 in order to
update the corresponding data in the memory 2. Thus the
snooping processor has to switch off the corresponding
change bit since the line of data which is stored in the

- 16 -

private cache of the snooping processor does not equal
the data having the same logical address i which are
stored in the memory 2. Furthermore the snooping
processor switches on its multiple copy bit to be
logically 1. This is because the changed data which is
outputted from the private cache of the snooping
processor in order to update the main memory 2 is
inputted into the private cache of the requesting
processor P1. As a consequence the corresponding multiple
copy bit in the storage space 7 of the private cache 4 of
the requesting processor P1 is also switched on to be
logically 1. Additionally, the changed data are put into
the shared cache memory and its V-Bit and MC-Bit are set
to 1.

In the following examples 5. to 8. the requesting
processor requires the line of data having the address i
because the requesting processor wants to store data into
the line of data having the logical address i. This is
called a "line fetch due to store". The example 5. is not
different from the example 1. as regards the status
information of the requesting and the snooping processor
apart from the fact that the change bit belonging to the
line of data having the address i which inputted into the
private cache 4 of the processor P1 is set to be
logically 1. This is because the processor P1 intends to
store new data into this line of data having the address
i.

Thereby it is anticipated that the line of data which is
inputted from the system bus 1 into the private cache 4
of the processor P1 is going to be changed by a storage
operation of the processor P1 into a storage line of the
storage array 6 into which this line of data is inputted.
The example 6. differs from the example 5. in that the
initial status information of the line of data considered

here in the snooping processor indicated that the line of
data having the address i which is stored in the private
cache of the snooping processor is valid (V-bit = 1) and
that the multiple copy bit and the change bit both equal
logically 0. Since only one of the processors can have a
valid, changed line of data stored in its private cache,
this line of data has to be invalidated in the snooping
processor.

In the example 7 the multiple copy bit equals logically 1
in the initial state of the snooping processor. For the
same reason as in the example 6. this line of data is
invalidated in the private cache of the snooping
processor. In the example 8. the private cache of the
snooping processor initially holds valid, changed line of
data having the address i. This situation is analogous to
the initial situation in the example 4. Again the
snooping processor performs a cast-out action in order to
update the main memory 2. The updated data is also
inputted into the private cache 4 of the processor P1 via
the system bus 1 as it is the case in the example 4.
However, as opposed to the example 4., the line of data
having the address i is invalidated in the private cache
of the snooping processor because these data are going to
be changed by the requesting processor. Again this is
reflected by the corresponding status information in the
private cache 4 of the requesting processor P1 where the V-
 and C-bits are logically 1 and the MC-bit is logically
0. This indicates that the updated data which are
inputted into the private cache 4 of the requesting
processor P1 via the system bus 1 from the snooping
processor which has performed the cast-out is going to be
changed by the requesting processor P1. This is
anticipated by the status information which indicates
that this line of data which is inputted in the private
cache 4 of the processor P1 is exclusive for the

- 18 -

processor P1. Additionally, the shared cache memory will
also set its V-Bit to 0.

The above described cache coherency scheme according to
the MESI-protocol is reflected in the signal diagram of
Fig.2 which illustrates the bus protocol. Fig.2 shows an
example of the bus snooping timing for a bus operation
like line fetch. It is assumed that one of the processors
of the multiprocessor system - for example processor P1 -
has a cache miss in its private cache memory and requires
a line of data having the address i to be inputted into
its private cache memory. Therefore the requesting
processor puts a corresponding command Cmd on the system
bus 1 in order to communicate its request to the other
bus participants. The requesting processor also outputs
the address i of the required line of data onto the
system bus 1 in order to specify which line of data is
needed. The fetch command Cmd as well as the address i
Adr are outputted onto the system bus 1 concurrently in
cycle 1. by the requesting processor. Concurrently
therewith the requesting processor raises the signal line
16 in order to "wake up" the other processors of the
system as well as the shared cache memory 3. This is to
initiate the bus snooping of the other processors and of
the shared cache memory 3. In the following it is assumed
that the shared cache memory 3 does not have a valid copy
of the requested data so that only the signals relating
to the private caches of the snooping processors are
shown in Fig.2.

The signal Cmd/Adr which is outputted by the requesting
processor onto the system bus 1 propagates as an
electromagnetic wave along the system bus 1. Due to the
electrical characteristics of the system bus 1 the signal
Cmd/Adr remains on the system bus 1 for two cycles. This
is necessary to support a near end reception from

processor to processor. A snooping processor which is
situated at the near end - or with other words - a
snooping processor neighboring the requesting processor
can only receive the signal Cmd/Adr when the reflected
electromagnetic wave comes back from the far end of the
system bus 1. This is because the voltage level of the
signal Cmc/Adr as it is issued by the requesting
processor is only half a full voltage swing from
logically 0 to logically 1. Hence a superposition of the
original signal Cmd/Adr and its reflected electromagnetic
wave is required in order to produce a voltage level of
the resulting signal at the near end of the system bus 1
which is sufficient to be detected by the near end
snooping processor. The timing which is shown in Fig.2
belongs to a snooping processor at the near end, for
example processor P2. The near end snooping processor
receives the signal Cmd/Adr in the cycle 3. in its bus-in
register (Bus-in Reg). This reception of the signal
Cmd/Adr is initiated by the signal CmdSel on the signal
line 16. As soon as the signal Cmd/Adr is received by the
snooping processor a directory search is carried out in
the directory of the private cache of the snooping
processor. This is to check whether there is a match of a
logical address stored in the directory and of the
logical address i of the requested line of data. If such
an address match is found in the directory and if the
valid bit (V-bit) of the corresponding storage line in
the private cache of the snooping processor considered
here is logically 1 there is a hit situation. This
situation is signaled to the other processors and to the
shared cache memory 3 via the signal line 17.

The snooping processor which has found the hit in the
directory of its private cache memory issues a signal
Hit-out in the cycle 4. The signal Hit-out is received by
the requesting processor one cycle later in the cycle 5.

- 20 -

The received signal is named Hit-in the reception of the
signal Hit-in informs the requesting processor about the
existence of a copy of the requested line of data in the
private cache memory of at least one of the snooping
processors. The signal Hit-in can only be received by the
requesting processor during the "Hit-window". In this
case the Hit-window is predefined as the cycle 5. The
resulting actions of both processors are described in the
above table 1. If no cast-out action is necessary for the
snooping processor, the snooping processor considered
here has just to turn on the corresponding multiple copy
bit to logically 1. In this case the signal Hit-out is
only issued for one cycle, i.e. cycle 4., by the snooping
processor on the signal line 17. This is the situation
shown in Fig.2.

If however a cast-out action is necessary this is
signaled to the other bus participants by the snooping
processor by issuing the signal Hit-out for more than one
cycle. This results in the appropriate actions which are
taken by the requesting and snoo-ping processor as
already explained with reference to table 1.

The shared cache memory 3 of the multiprocessor system
shown in Fig.1 operates in a "store thru mode". This
means that an update of a line of data in the main memory
2 is written into the shared cache memory 3 concurrently
with the update of the main memory 2. The shared cache
memory 3 has a much shorter access time than the main
memory 2 which contributes to a more efficient usage of
the system bus and thereby to a faster operation of the
micro processor system. The valid and multiple copy bits
which are stored in the storage space 13 of the shared
cache memory 3 are controlled in the same way as the
corresponding status bits in the private caches according
to the MESI-protocol. If the valid bit (V-bit) of a line

of data which is stored in the storage array 12 is
logically 1 this indicates that this line of data is a
valid copy of the same line of data having the same
address i which is stored in the main memory 2.

If the multiple copy bit (MC-bit) of a valid line of data
having the address i is logically 1 this indicates that
there is a valid copy of this line of data in at least 2
of the private caches of 2 different processors of the
multi processor system. The provision of the multiple
copy bit in the storage space 13 of the shared cache
memory 3 is essential for a further improvement of the
usage of the system bus 1.

In the following an example of the operation of the
shared cache memory 3 is given:

In the initial situation it is assumed that one of the
processors, for example processor P1 requires a line of
data having the address i. This line of data is not
present in the private cache 4 of the processor P1 so
that there is a cache miss in the private cache 4. As a
consequence the processor P1 puts a line fetch command
together with the address i Cmd/Adr on the system bus 1
(cf. Fig.1). It is assumed that the requested line of
data is neither stored in any of the private caches of
the other processors nor in the storage array 12 of the
shared cache memory 3. This means that the requested line
of data is only present in the main memory 2 of the
microprocessor system. As a consequence the requested
line of data is outputted from the main memory 2 onto the
system bus 1 and inputted into the private cache 4 of the
requesting processor P1. Concurrently to the inputting of
the requested line of data into the private cache 4 of
the requesting processor P1 the requested line of data is
also inputted into the shared cache memory 3 and stored

in one of the storage lines of the storage array 12.

This storage operation in the shared cache memory 3 is
carried out under the control of the logic 15. The valid,
multiple copy and change bit of the requested line of
data which is stored now in the storage array 6 of the
microprocessor P1 is 1,0,0, respectively. The valid bit
of the requested line of data which is stored in the
storage array 12 of the shared memory 3 is also logically
1 and the corresponding multiple copy bit is also
logically 0. The valid and the multiple copy bit of the
requested line of data which is stored now in the storage
array 12 of the shared memory 3 are stored in storage
fields of the storage space 13 which belong to the
storage line in the storage array 12 in which the
requested line of data is stored now.

It is assumed that a second request for the same line of
data having the address i comes from another processor,
for example processor P2. The second request is also due
to a cache miss in the private cache 5 of the requesting
processor P2. Again, the requesting processor P2 outputs
a fetch command and the corresponding address i Cmd/Adr
onto the system bus 1. This causes the processor P1 to
signal a Hit-situation (cf. Fig.2). The hit is signaled
to the other processors and to the shared cache memory 3
via the signal line 17 by the processor P1.

The shared cache 3 has also a hit because the requested
line of data having the address i is stored in the
storage array 12 and is valid. The shared cache memory 3
waits until the window cycle (cf. cycle 5. in Fig.2) is
over. If there is no Hit-out signal which remains active
for more than one cycle this means that there is no cast-
out situation. This is the case in the example considered
here because the processor P1 has not changed the valid

copy of the line of data having the address i which has
been inputted via the system bus 1 from the main memory
2. The Hit-situation in the private cache 4 of the
processor P1 does only cause that the multiple copy bit
in the storage space 7 which belongs to the storage line
of the storage array 6 in which the line of data having
the address i is stored belongs. After the window cycle
is over and if no cast-out occurred the shared cache
memory 3 outputs the requested line of data onto the
system bus 1. This is carried out under the control of
the logic 15. The requested line of data is inputted into
the private cache 5 of the requesting processor P2.

The multiple copy bit in the storage field of the storage
space 13 which belongs to the line of the storage array
12 in which the requested line of data is stored is set
to logically 1 under the control of the logic 15. This is
because there is a valid copy of this line of data in
both of the private caches 4 and 5 of the processor P1
and P2. The corresponding multiple copy bit in the
storage space 9 of the requesting processor P2 is also
turned on correspondingly.

Furthermore, it is assumed that a third processor, for
example processor P3 which is not shown in Fig.1,
requests the same line of data having the address i.
Again this causes Hit-situation in the shared memory 3
which has stored the requested line of data in its
storage array 12. The respective status information
stored for this line of data in the storage space 13
indicates that this line of data which is stored in the
storage array 12 is valid and that there are 2 ore more
valid copies of this line of data in different private
caches of different processors in the system. By
definition this situation can only occur if the requested
line of data is not exclusive to one of the processors.

Such a situation is not allowed because in this case the
consistency of the data stored in the different cache
memories could no longer be maintained.

As a consequence it is excluded by the MESI-protocol -
which is employed in this preferred embodiment of the
invention to maintain data consistency - that there is an
exclusive line of data having the address i in one of the
private caches of the processors and that the multiple
copy bit for this line of data is logically 1 in at least
another one of the private caches or in the storage space
13 of the shared cache memory 3 at the same time. Since -
by definition - the requested line of data having the
address i which is requested by the processor P3 is not
an exclusive line of data of any of the processors
because the corresponding multiple copy bit in the
storage space 13 of the shared memory 3 is logically 1,
no cast-out situation can occur.

As a consequence the shared cache memory 3 does not have
to wait until the window cycle (cf. cycle 5. in Fig.2) is
over. Hence, the shared memory 3 outputs the requested
line of data directly after the reception of the request
onto the system bus 1 from where it is inputted into the
private cache of the requesting processor P3.

The same process is carried out analogously if there are
further requests from other processors for this line of
data later on. If one of the processors which has a valid
copy of this line of data once to store into this line,
the processor must issue a line invalidate command prior
to the storage operation, again by definition. This
causes an invalidation of all copies of this line of data
in all of the caches, including the shared cache memory 3
and also the reset of all corresponding multiple copy
bits of this line of data having the address i to

logically 0.

In a microprocessor system there are often software
objects which are frequently read by the processors and
which remain unchanged for a long time or even for ever.
Such software objects typically consist of code. The
background is that in a lot of nower days microprocessor
architectures the instruction set is predefined and an
instruction can not be changed as such.

In the following it is assumed that the processor P1
requests a line of data having the address j whereby the
requested data are instructions. The corresponding
command which is issued by the processor is called "line
fetch due to instruction fetch". Thereby the other bus
participants are informed that the requested line fetch
is carried out for the provision of a line of
instructions having the address j to the requesting
processor P1. As in the example considered above it is
assumed in the initial situation that the required line
of data is not present in any of the caches of the
microprocessor system. As a consequence the requested
line of data is outputted by the main memory 2 onto the
system bus 1 and inputted both into the private cache 4
of the requesting processor P1 and into the shared cache
memory 3. As opposed to the normal case considered above
the multiple copy bit for this line of data having the
address j is turned on immediately in the storage space
13 even though there is no further copy of this line of
data in any other of the private caches of the other
processors. This is because it is relatively unlikely or
even impossible that this line of data is going to be
changed by the requesting processor P1. As a consequence
it is assumed that the requested line of data will not be
an exclusive line of data for the processor P1. This
makes it possible to set the multiple copy bit of the

requested line of data having the address j in the
storage space 13 of the shared cache memory 3 already to
logically 1 at the first request of this line by one of
the processors, in this example by the processor P1.
Consequently, the MC-Bit of the requesting processor must
be set to 1.

If a second request for the same line of data having the
address j occurs now for example from the processor P2
this has the advantageous that the shared cache memory 3
does not have to wait until the window cycle is over.
Since it is already clear from the beginning - as it is
indicated by the multiple copy bit which is logically 1 -
that no cast-out situation can occur the requested line
of data having the address j is directly outputted from
the shared cache memory 3 after receipt of the
corresponding request from the processor P2 by the shared
cache memory 3. Again this is carried out under the
control of the logic 15.

Even though it is unlikely it may happen that the
processor P1 wants to store into the line of data having
the address j prior to the second request of the
processor P2. In this case the processor P1 has to issue
a command on the system bus 1 so that the corresponding
line of data which is stored in the storage array 12 of
the shared cache memory 3 is invalidated prior to the
storage operation of the processor P1. The invalidation
of this line of data in the shared cache memory 3 is
carried out under the control of the logic 15.

The signal diagram shown in Fig.3 illustrates the
enhancement of the performance of the multiprocessor
system which is due to the shared cache memory 3. The
situation shown in Fig.3 corresponds to the situation as
shown in Fig.2 with respect to the requesting processor

and the snooping processors. In addition to Fig.2 the
signals of the shared cache memory 3 ("shared cache") are
illustrated in Fig.3. The shared cache memory 3 is placed
at the far end of the system bus 1. Since the far end of
the bus 1 has a high impedance a command on the bus
(Cmd/Adr) is received by the shared cache memory 3
already when the corresponding electromagnetic signal
reaches the far end. As a consequence the time for the
reception of the signal is shorter for the shared cache
memory 3 as compared to a near end processor which has to
wait for the reflection of the electro magnetic signal,
as already explained above.

Since the shared cache memory 3 is placed at the far end
of the system bus 1 the electromagnetic signal from the
requesting processor (Cmd/Adr) is already received in
cycle 2. in the bus-in register of the shared cache
memory 3. Thereby an directory search in the directory 14
of the shared cache memory 3 is initiated. In the example
considered here there is an address match in the
directory 14. Furthermore it is assumed that the
corresponding line of data which is stored in the shared
cache memory 3 is valid and that the corresponding
multiple copy bit of this line of data is logically 1.

In response to this the storage array 12 where the
requested line of data is stored is accessed under the
control of the logic 15. This line of data is outputted
via the data-out register ("Data-outReg") onto the system
bus 1 in four "shots" of data in the cycles 3., 4., 5.
and 6. As a result the output operation of the requested
data onto the bus is already accomplished in the cycle 6.
This is made possible because of the multiple copy bit in
the shared cache memory 3 which allowed to directly
output the data on the system bus 1 without having to
wait for the window cycle (cycle 5.). If the shared cache

memory 3 would have to wait for the window cycle the
output of the requested line of data could only begin in
cycle 6. and would finish in cycle 9. Hence, the result
is a performance enhancement of three cycles which is
due to the teaching of the invention.

C L A I M S

1. A cache memory device for usage as a shared cache in
    a multiprocessor system, said cache memory device
    being organized in a plurality of storage blocks
    (line 1, line 2,..., line m) and having storage
    means (13) for storage of status information (V, MC-
    bit) for each of said storage blocks, said status
    information being indicative of valid and/or shared
    data of said storage blocks, said cache memory
    device further comprising logic means (15) to
    selectively store data in one said storage blocks
    and to set said status information correspondingly.

2. The cache memory device according to claim 1 said
    logic means being adapted to store data in one of
    said storage blocks, if valid data is fetched from
    one of the processors (P1, P2) of said
    multiprocessor system from a main memory (2) or if
    changed data is cast out by one of said processors.

3. The cache memory device according to claim 1 or 2
    said status information comprising a valid bit (V-
    bit) and a multiple copy bit (MC-bit) for each of
    said storage blocks.

4. A multiprocessor system comprising a plurality of
    processors (P1, P2), each of said processors having
    a private cache memory device (4, 5) associated
    thereto, said multiprocessor system further
    comprising a main memory (2) and a shared cache
    memory device (3) according to any one of the
    preceding claims.

5. The multiprocessor system of claim 4 each of said
    private cache memory devices having means (7, 9) for

storage of status information (V, MC, C-bit)
indicative of valid, shared, exclusive or modified
data being stored therein.

6.   The multiprocessor system of claim 5 said means for
      storage comprising a set of three bits for each of
      the storage blocks of one of said private caches,
      said set consisting of a valid bit (V-bit), a
      multiple copy bit (MC-bit) and a change bit (C-bit).

7.   The multiprocessor system according to any one of
      the claims 4 to 6 said logic means being adapted to
      cause the output of data of one of said storage
      blocks of said cache memory device in response to a
      corresponding request of one of said processors on
      condition that said status information of that
      storage block having the requested data stored
      therein indicates that said requested data is valid
      and shared irrespective of a hit situation occurring
      in one of the private caches of the other
      processors.

8.   The multiprocessor system according to claim 7 said
      logic means being adapted to cause the output of
      said requested data prior to the signalling of a
      said hit situation occurring in one of the private
      caches of the other processors.

9.   The multiprocessor system according to any one of
      the claims 4 to 8 said logic means being adapted to
      set said status information being indicative of
      shared data in the case that data which generally
      remains unchanged, such as instructions, is
      requested by one of said processors irrespective of
      said requested data being shared by one of the
      private caches of the other processors.

10. A computer system comprising a multiprocessor system according to any one of the claims 4 to 9.

11. A method to operate a shared cache memory device (3) in a multiprocessor system; said cache memory device being organized in a plurality of storage blocks (line 1, line 2, ... , line m), said system having a plurality of processors (P1, P2) each of which having a private cache memory device (4, 6) associated thereto, comprising the steps of:

   a) storage of status information (V, MC-bit) for each of said storage blocks, said status information being indicative of valid and/or shared data of said storage blocks;

   b) outputting shared data being stored in one of said storage blocks of said shared cache memory device in response to a corresponding request of one of said processors irrespective of a hit situation occurring in one of the private caches of the other processors.

12. The method according to claim 11 said outputting of shared data beginning prior to the signalling of a said hit situation occurring in one of the private caches of the other processors.

13. The method according to claim 11 or 12 comprising a step of storage of information indicative of the presence of shared data even if the corresponding data to be stored in said storage block is not shared on condition that said data to be stored generally remains unchanged, such as instructions.

Fig. 1
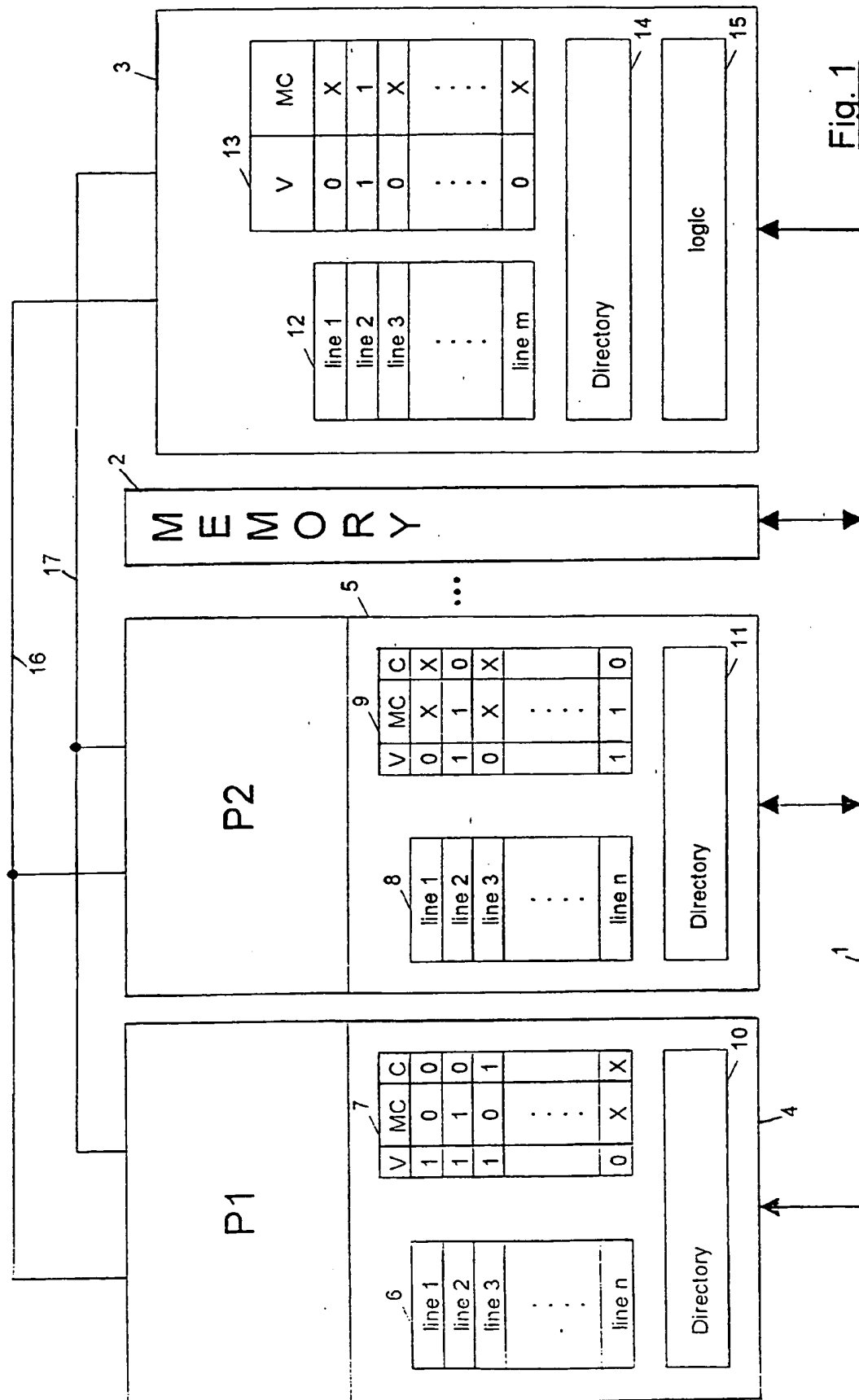
Cycle	1.	2.	3.	4.	5.	6.
Requesting processor:						
Adr/Data bus	Cmd/ Adr	Cmd/ Adr				
Cmd Sel						
Hit - in						
Private cache of snooping processor		Unused	Cmd/ Adr			
Bus-in Reg						
Dir search						
Hit - out						
Hit window						

Fig. 2

Cycle	1.	2.	3.	4.	5.	6.	7.	8.	9.
Requesting processor:	Cmd/ Adr	Cmd/ Adr							
Adr/Data bus									
Cmd Sel									
Hit - in									
Private cache of snooping processor		Unused	Cmd/ Adr						
Bus-in Reg									
Dir search									
Hit - out									
Hit window									
Shared cache:		Cmd/ Adr	Cmd/ Adr						
Bus-in Reg									
Dir search / MC bit									
Cache access									
Data-out Reg									

Fig. 3

INTERNATIONAL SEARCH REPORT

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6   G06F12/08

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 6   G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	EP,A,0 608 638 (IBM) 3 August 1994 see abstract see column 4, line 16 - column 7, line 4; figures 1,4 ---	1-8, 10-13
A	MICROPROCESSOR AND MICROPROGRAMMING 32, NORTH-HOLLAND, vol. 32, no. 1-5, 1991 pages 215-220, DOETTLING 'Data consistency in a multiprocessor system with 'store in' cache concept' cited in the application see the whole document ---	1-6,11

-/--

[X] Further documents are listed in the continuation of box C.   [X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
26 March 1996	22. 04. 96

| Name and mailing address of the ISA
European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (- 31-70) 340-2040, Tx. 31 651 epo nl,
Fax (- 31-70) 340-3016 | Authorized officer

Nielsen, O |

Form PCT/ISA-210 (second sheet) (July 1992)

page 1 of 2

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT		
Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	EP,A,0 481 233 (IBM) 22 April 1992 see column 3, line 17 - column 4, line 34; figures 6,15 ---	1-6,11
A	IBM TECHNICAL DISCLOSURE BULLETIN, vol. 27, no. 1A, June 1984 NEW YORK, US, pages 298-300, ANONYMOUS 'Second Level Cache for MP Systems' see the whole document -----	1,2,4,7, 8,11,12

1

Inte. .onal Application No
PCT/EP 95/02847

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
EP-A-608638	03-08-94	CA-A-	2113867	21-07-94
		CN-A-	1092883	28-09-94
		JP-A-	6282528	07-10-94
EP-A-481233	22-04-92	JP-A-	4230549	19-08-92